

חזרה בנושא מיון מיזוג של רשימות

מיון מיזוג של רשימות

מטרות

תרגול השימוש באוסף הכללי רשימה
תרגול גם בנושאי אלגוריתמיקה ורקורסיה

רמת השאלה

בינונית-גבוהה

מה עליכם לעשות?

א. לפניכם הפעולה mergeLists המקבלת שתי רשימות של מספרים שלמים ממוינות בסדר עולה. הפעולה מחזירה רשימה שהיא מזוג של שתי הרשימות שהתקבלו.

בגוף הפעולה mergeLists חסרים 6 ביטויים הממוספרים (1) - (6). עליכם להשלים אותם.

```
public static List<Integer> mergeLists (List<Integer> list1,
                                       List<Integer> list2)
{
    Node<Integer> pos1 = list1.getFirst();
    Node<Integer> pos2 = list2.getFirst();
    Node<Integer> pos3 = null;
    List<Integer> list3 = new List<Integer>();

    while( _____ (1) _____ )
    {
        int small;
        if(pos1.getInfo() < pos2.getInfo())
        {
            small = pos1.getInfo();
            pos1 = _____ (2) _____;
        }
        else
        {
            small = _____ (3) _____;
            pos2 = pos2.getNext();
        }
        pos3 = _____ (4) _____;
    }
    while(pos1!=null)
    {
        pos3 = _____ (5) _____;
        pos1 = pos1.getNext();
    }

    while( _____ (6) _____ )
    {
        pos3 = list3.insert(pos3, pos2.getInfo());
        pos2 = pos2.getNext();
    }

    return(list3);
}
```

ב. נתחו את יעילות הפעולה mergeLists.

ג. לפניכם הפעולה splitList(...) המפצלת רשימה שהתקבלה כפרמטר, לשתי רשימות שוות באורכן (כמעט). אם הרשימה המתקבלת מכילה מספר אי-זוגי של איברים, אז ברשימה הראשונה יהיה איבר אחד פחות מהרשימה השנייה. הפעולה שומרת על סדר האיברים ברשימה המקורית. הפעולה מחזירה את שתי הרשימות מאוחסנות במערך חד-ממדי שבו שני תאים. בכל אחד מהתאים תאוחסן מחצית הרשימה המקורית.

לדוגמה, אם הרשימה המתקבלת היא (משמאל לימין): 12, 3, 4, 23, 11, 5, 8, 3, 2, 16, 45, 22, 1

אז שתי הרשימות שיוחזרו יהיו: הראשונה: 12, 3, 4, 23, 11, 5 והשנייה: 8, 3, 2, 16, 45, 22, 1

הערה: הפעולה splitList(...) עוברת על הרשימה פעם אחת בלבד באמצעות שתי הפניות pos1 ו-pos2 המתחילות מהאיבר הראשון ברשימה. ההפניה pos1 מתקדמת צעד אחד בכל שלב ואילו ההפניה pos2 מתקדמת בדילוגים (בכל פעם שני צעדים).

בגוף הפעולה splitList(...) חסרים 6 ביטויים הממוספרים (1) - (6). עליכם להשלים אותם.

```
public static List<Integer>[] splitList(List<Integer> list)
{
    List<Integer>[] split = new List[2];

    split[0] = _____(1)_____ ;
    split[1] = _____(2)_____ ;

    Node<Integer> pos1 = list.getFirst();
    Node<Integer> pos2 = list.getFirst();
    Node<Integer> pos3 = null;

    while (pos2 != null && pos2.getNext() != null)
    {
        pos3 = _____(3)_____ ;
        pos1 = pos1.getNext();
        pos2 = _____(4)_____ ;
    }

    _____(5)_____ ;
    while ( _____(6)_____ )
    {
        pos3 = split[1].insert(pos3, pos1.getInfo());
        pos1 = pos1.getNext();
    }

    return split;
}
```

ד. נתחו את יעילות הפעולה splitList.

ה. לפניכם הפעולה mergeSort(...) המממשת את האלגוריתם "מיון-מיוזג" בצורה רקורסיבית; הפעולה מקבלת רשימת מספרים לא ריקה ומחזירה רשימה ממוינת בסדר עולה של כל המספרים שהיו ברשימה המקורית. הפעולה אינה משנה את הרשימה המקורית.

בגוף הפעולה mergeSort(...) חסרים 4 ביטויים הממוספרים (1) - (4). עליכם להשלים אותם תוך שימוש בפעולות המופיעות בסעיפים א' ו-ג' לעיל.

```
public static List<Integer> mergeSort(List<Integer> list)
{
    List<Integer> sortedList;

    if(list.getFirst().getNext() == null) // אם קיים איבר אחד ברשימה
    {
        sortedList = new List<Integer>();
        sortedList.insert(null, list.getFirst().getInfo());
    }
    else
    {
        List<Integer>[] split = _____ (1) _____;
        List<Integer> left = _____ (2) _____;
        List<Integer> right = _____ (3) _____;
        sortedList = _____ (4) _____;
    }

    return(sortedList);
}
```

הנחיות מיוחדות

- א. שאלה זו בודקת את יכולת התלמיד לעבוד עם האוסף הכללי רשימה המוגדר על ידי המחלקה List הגנרית ותוך שימוש במחלקה Noed לצורך מעבר על הרשימה.
- ב. תרגיל זה בודק גם את היכולת האלגוריתמית של התלמיד וניתוח יעילות.
- ג. הבקשה: נתחו יעילות – יכולה להיענות בכמה דרכים שכולן טובות. התלמיד יכול להחזיר פונקציה זמן ריצה מדויקת, כך תוכלו לעקוב שהוא ניתח נכון מה קורה בפעולה. התלמיד יכול להחזיר תשובה תמציתית של סדר הגודל של יעילות הפעולה אך אז עליו לנמק ולהסביר איך הגיע לסדר גודל זה. בכל דרך שהתלמיד יבחר לפתור את השאלה – יש להקפיד על כך שיהיה ברור שהתלמיד מבין כיצד הגיע לחישוב היעילות.
- ד. בסעיף ה' של השאלה, הסעיף הקשה יותר, מדובר בפעולה רקורסיבית מורכבת שנעזרת גם בפעולות שנכתבו בסעיפים קודמים של השאלה. ניתן להקל בסעיף זה על תלמידים חלשים, לתת את פתרון הסעיף ולבצע עליו מעקב רקורסיבי. כך יקבלו התלמידים מושג על מהות הרקורסיה לבעיה פשוטה יחסית.
- ה. יש לשקול להציג את הפתרון הרקורסיבי מול הפתרון האיטרטיבי.

הערה: כפי שניתן לראות שאלה זו בנויה על השלמת קוד. במבט ראשון ייתכן ושאלות מסוג זה נראות קלות, אך לא כך הדבר! בפתרון שאלה מסוג זה יש צורך "להיכנס לראש" של המתכנת שכתב את הפעולות ולנסות לבדוק למה "התכוון המשורר?" ורק אז ניתן להשלים את הקוד. זה לא ממש קל, אבל לתלמיד ממוצע הפתרון אמור להיות לא מסובך.


```

public static List<Integer> mergeLists (List<Integer> list1,
                                         List<Integer> list2)
{
    Node<Integer> pos1 = list1.getFirst();
    Node<Integer> pos2 = list2.getFirst();
    Node<Integer> pos3 = null;

    List<Integer> list3 = new List<Integer>();

    while (pos1!=null && pos2!=null)
    {
        int small;
        if (pos1.getInfo() < pos2.getInfo())
        {
            small = pos1.getInfo();
            pos1 = pos1.getNext();
        }
        else
        {
            small = pos2.getInfo();
            pos2 = pos2.getNext();
        }
        pos3 = list3.insert (pos3, small);
    }

    while (pos1!=null)
    {
        pos3 = list3.insert (pos3, pos1.getInfo());
        pos1 = pos1.getNext();
    }

    while (pos2!=null)
    {
        pos3 = list3.insert (pos3, pos2.getInfo());
        pos2 = pos2.getNext();
    }

    return (list3);
}

```

סעיף ב: זמן הריצה של הפעולה mergeLists הוא $O(n)$ לינארי, כאשר אורך הקלט n מוגדר כך $n=n_1+n_2$.
 n_1 הוא מספר האיברים ברשימה $list_1$ ו- n_2 הוא מספר האיברים ברשימה $list_2$.

```

public static List<Integer>[] splitList (List<Integer> list)
{
    List<Integer>[] split = new List[2];

    split[0] = new List<Integer>();
    split[1] = new List<Integer>();

    Node<Integer> pos1 = list.getFirst();
    Node<Integer> pos2 = list.getFirst();
    Node<Integer> pos3 = null;

```

```

while(pos2!=null && pos2.getNext()!=null)
{
    pos3 = split[0].insert(pos3,pos1.getInfo());
    pos1 = pos1.getNext();
    pos2 = pos2.getNext().getNext();
}

pos3 = null;
while(pos1 != null)
{
    pos3 = split[1].insert(pos3,pos1.getInfo());
    pos1 = pos1.getNext();
}

return(split);
}

```

סעיף ד: זמן הריצה של הפעולה `splitList` הוא $O(n)$ ליניארי, כאשר אורך הקלט n הוא מספר האיברים

ברשימה `list`.

סעיף ה:

```

public static List<Integer> mergeSort(List<Integer> list)
{
    List<Integer> sortedList;

    if(list.getFirst().getNext() == null) // אם ברשימה יש איבר אחד
    {
        sortedList = new List<Integer>();
        sortedList.insert(null, list.getFirst().getInfo());
    }
    else
    {
        List<Integer>[] lst = splitList(list);
        List<Integer> left = mergeSort(lst[0]);
        List<Integer> right = mergeSort(lst[1]);
        sortedList = mergeLists(left, right);
    }

    return(sortedList);
}

```

מקור השאלה

אילן פרץ, מרכז מגמת מחשבים באורט גבעת רם, ירושלים.